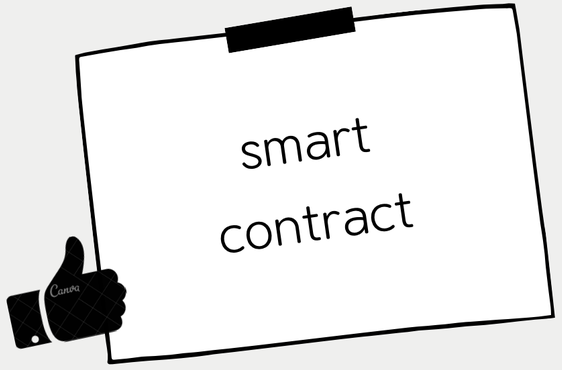
허정원

스마트 컨트랙트



목차

선정 이유

- ✅ 자동화된 계약시스템
- ✅ 탈중앙화
- ✅ 인위적인 변경 불가능

Why it was chosen

Smart contract

블록체인 기술을 기반으로
자동화된 계약시스템을 구현하는 프로그램

중개자 없이 계약의 조건이 충족되었을 때
자동으로 실행되는 방식으로 동작

스마트 컨트랙트의 특징

about Smart Contract

1 smart contract

2 smart contract & database

3 application

1

자동화된 실행 🤖

- 조건이 충족되면 자동으로 실행됨

2

불변성 & 투명성 🔒

- 한 번 실행된 후에는 내용 변경 불가
- 누구나 계약 조건과 결과 검증 가능

3

탈중앙화 🌍

- 블록체인 네트워크 상에서 실행
- 중개자가 필요하지 않음

4

보안 🛡️

- 해킹과 위조가 어려움
- 코드가 잘못 작성된 경우 보안 취약점 발생

스마트 컨트랙트 동작 원리

How are you feeling today?

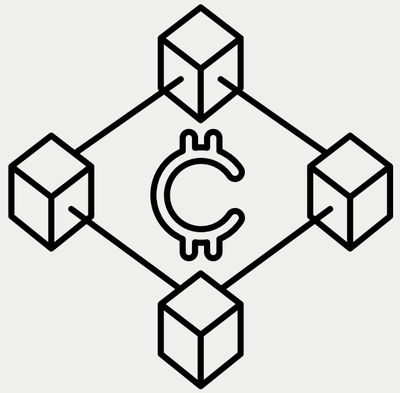
1 smart contract

2 smart contract & database

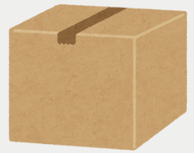
3 application

Solidity
이더리움 기반
코드 작성

1) 배포



5) 결과 기록



4) 실행

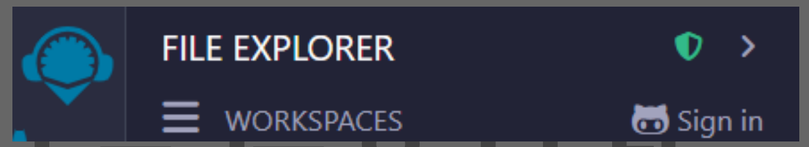
2) 조건 설정

A가 B에게 100 ETH를 송금하면, B는 상품을 발송한다.

A → 100ETH → B

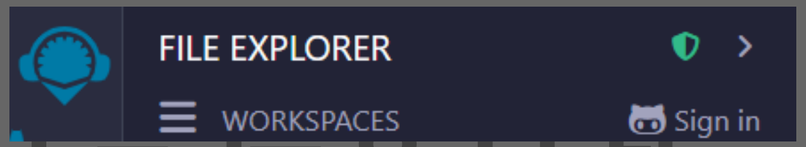
3) 트리거

Remix IDE



Home | vending.sol

```sol
// SPDX-License-Identifier: MIT License
pragma solidity >=0.7.0 <0.9.0; // 버전

contract VendingMachine {
    address owner; // owner 주소
    string[] itemList = ["Cider", "Juice", "Coke", "Water"];
    mapping(string => uint8) itemStock;

    // 스마트컨트랙트 외부에서 추적할 수 있도록 상태 변화 로그를 찍는 용도
    event SomeoneBuy(string itemName, uint8 stock);
    event SoldOut(string itemName);
    event AddItemStock(string itemName, uint8 stock);

    // 계약이 배포될 때 자동 실행
    constructor() {
        owner = msg.sender; // 배포자

        // 초기 수량 저장
        for (uint8 i = 0; i < itemList.length; i++) {    infinite gas 788600 gas
            itemStock[itemList[i]] = 10;
        }
    }

    // owner만 실행 가능한 함수 제한자
    modifier onlyOwner() {
        require(owner == msg.sender, "Only Owner!!");
         ;
```



Remix IDE

```
// 구매할 때 실행되는 함수
function buyItem(uint8 _index) public payable returns (string memory) {
    require(msg.value == 1 ether, "price 1 ETH per piece");

    if (itemStock[itemList[_index]] > 0) {    infinite gas
        // 1ETH 보내야만 구매 가능
        itemStock[itemList[_index]]--; // 재고 감소
        emit SomeoneBuy(itemList[_index], itemStock[itemList[_index]]); // 이벤트 발생
    } else if (itemStock[itemList[_index]] == 0) { // 품절
        emit SoldOut(itemList[_index]);
        revert("Sold Out"); // 트랜잭션 취소
    }

    return "Success!";
}

// 재고 추가
function addStock(uint8 _index, uint8 _num)
    public
    onlyOwner // owner만 실행 가능
    returns (uint8 currentStock)
{
    itemStock[itemList[_index]] += _num;
    currentStock = itemStock[itemList[_index]];    infinite gas

    emit AddItemStock(itemList[_index], currentStock);

    return currentStock;
}
```

스마트 컨트랙트 동작 원리

How are you feeling today?

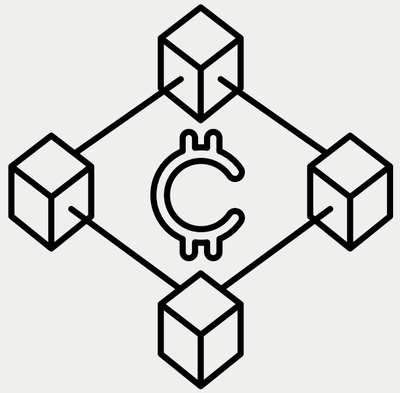
1 smart contract

2 smart contract & database

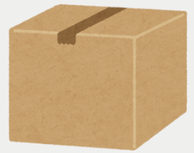
3 application

Solidity
이더리움 기반
코드 작성

1) 배포



5) 결과 기록



4) 실행

2) 조건 설정

A가 B에게 100 ETH를 송금하면, B는 상품을 발송한다.

A → 100ETH → B

3) 트리거

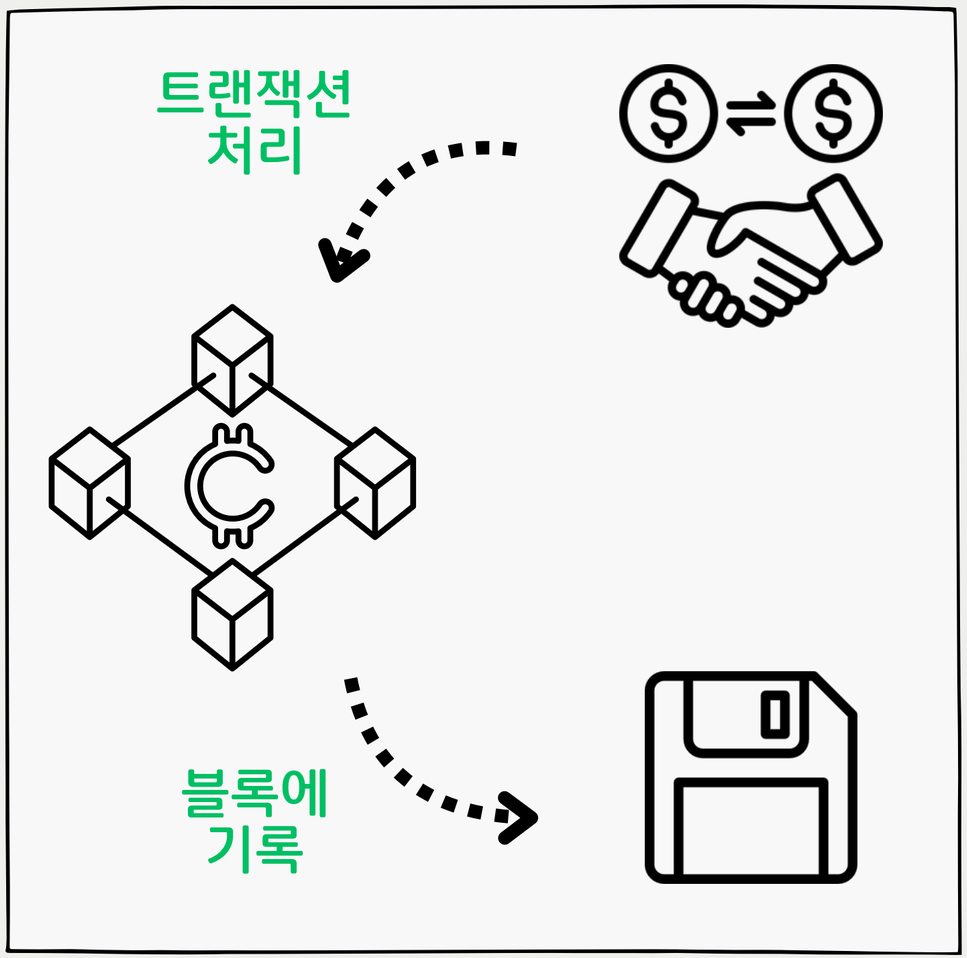
스마트 컨트랙트와 트랜잭션

smart contract and transaction

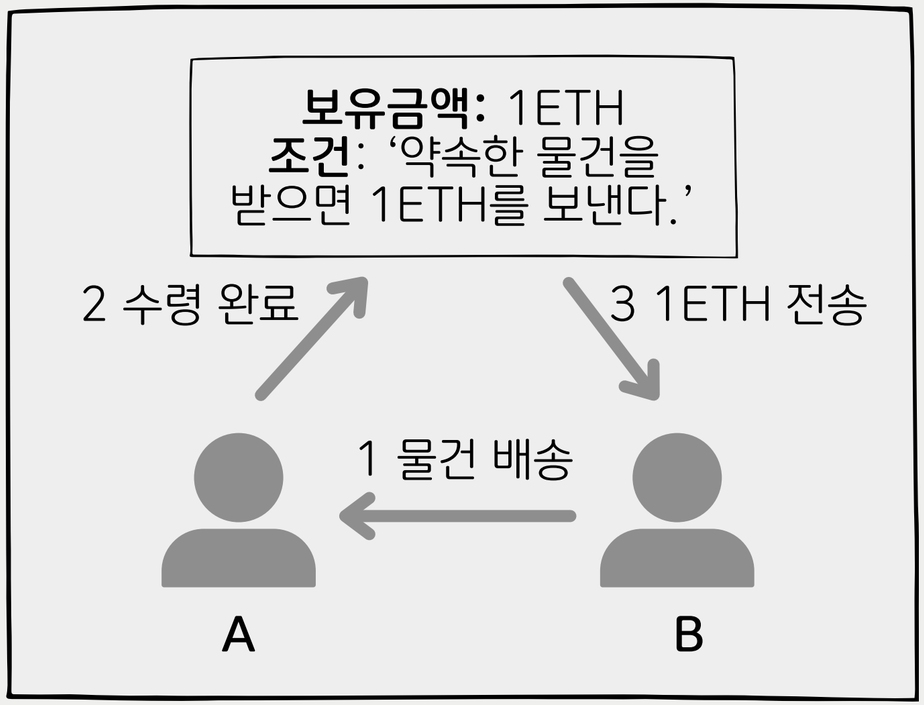
1 smart contract

2 smart contract & database

3 application



← 하나의 **분산 데이터베이스** 역할을 하게 됨



블록체인에 기록됨

트랜잭션 관점에서 스마트 컨트랙트와 DB

1 smart contract

2 smart contract & database

3 application

구분	스마트 컨트랙트 트랜잭션 🔀	데이터베이스 트랜잭션 💱
변경 가능 여부	✅ 변경 불가능 ❌	변경 가능 (Rollback, update) ⭕
처리 속도	느림 (블록체인 검증 필요) 🔻	✅ 빠름 (중앙 서버에서 즉시 실행) 🔺
처리 비용	Gas Fee(수수료) 발생	✅ 별도 비용 없음
검증 방식	네트워크 노드들이 합의	단일 서버(DBMS)에서 트랜잭션 관리
데이터 정합성	✅ 모든 노드가 동일한 상태 유지	✅ ACID 원칙 적용
롤백 가능 여부	실행 후 취소 불가능	✅ 롤백 가능

스마트 컨트랙트와 DB

smart contract and database

1 smart contract

2 smart contract & database

3 application

완전한 **투명성과 불변성, 자동화**가 필요하다면? → **스마트 컨트랙트**

빠른 데이터 처리와 변경이 필요하다면? → **데이터베이스**

↓

스마트 컨트랙트와 DB를 **상호보완적 관계**로도 사용 가능

스마트 컨트랙트와 DB

smart contract and database

1 smart contract

2 smart contract & database

3 application

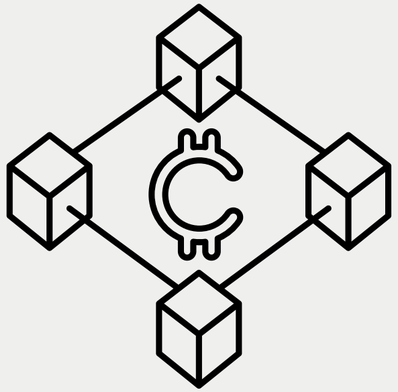
스마트 컨트랙트와 DB는 **상호보완적 관계**

데이터 조작 불가능

중개자 / 중앙서버 필요 ✕

전세계 누구나 사용

해킹에 강함



스마트 컨트랙트

데이터베이스

개인 정보 유출 우려 ▼

빠른 실시간 처리

Rollback 가능

API, DB 읽기 가능

스마트 컨트랙트와 DB 사례

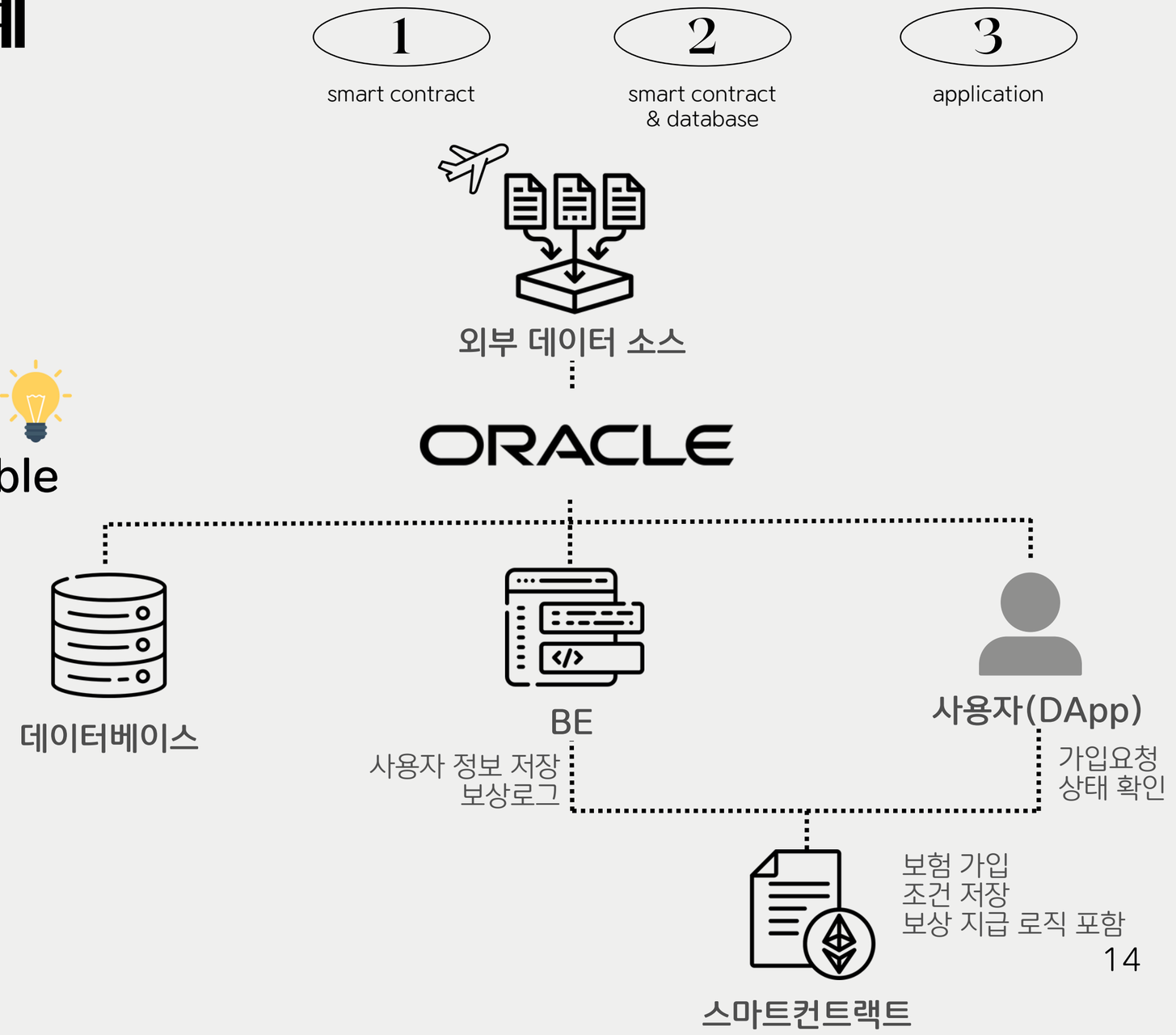
smart contract and database

ETHERISC

블록체인 기반 **탈중앙 보험 플랫폼**

Make Insurance Fair and Accessible

1. 사용자가 항공편 지연 보험 가입
2. 사용자 정보를 DB에 저장
3. 스마트컨트랙트에 가입 요청(트랜잭션 생성)
4. 오라클이 항공 API에서 지연 정보 수집
5. 지연 발생 시 오라클이 스마트컨트랙트 호출
6. 보험금 자동 지급
7. 지급 내역을 DB에 기록
8. 사용자 UI에 보상상태 업데이트



활용 분야

실제로 많이 활용되는 분야

1 smart contract

2 smart contract & database

3 application

금융

❗ 대출 플랫폼에서 담보 대출, 스테이킹, 자동 청산 등 구현

❗ DEX(탈중앙화 거래소)에서는
AMM(Automated Market Maker)으로 자동 거래 실행

📌 실제 사례
✓ **Uniswap** → 중개자 없이 토큰 스왑 거래 가능
✓ **Aave** → 담보 대출 자동 실행
✓ **MakerDAO** → 스테이블 코인(DAI) 발행 및 담보 대출

NFT & 게임

❗ NFT 시장에서의 핵심 기술

❗ 게임 내 아이템을 NFT로 발행
→ 유저 간 거래, 소유권 보장

📌 실제 사례
✓ **OpenSea** → NFT 거래
✓ **Axie Infinity** → 게임 캐릭터/아이템을 NFT로 거래
✓ **Decentraland** → 가상 부동산 소유권 관리

활용 분야

아직 활성화되지 않은 분야와 그 이유

1 smart contract

2 smart contract & database

3 application

보험

현실 적용이 어려움

- 복잡한 **법적 해설 & 예외 상황**
- 계약 조건이 정확한 데이터 입력을 전제로 함
- 사고나 질병을 데이터만으로 판단하기 어려움

Etherisc

날씨 기반 보험
폭우 발생, 자동 보험금 지급

공급망 관리

모든 거래 과정을 블록체인에 기록
→ **투명성**이 높아짐

데이터를 블록체인과 연결하는 것이 어렵고, 기존 시스템과 통합하는 비용 多

- SCM – 블록체인 동기화하는 **인프라 부족**
- 기존 **ERP시스템과의 연동**이 어려움

IBM Food Trust

월마트, 네슬레 등이 참여
식품 공급망을 블록체인으로 관리

부동산 거래

중개인을 줄이고 거래 비용 절감 가능

- 법적 문제와 기존 등기 시스템과의 통합
- 법적으로 블록체인 계약 인정 여부
- 부동산 등기 시스템이 **정부 DB**와 연결되어 있어 통합이 어려움
- 가격 협상, 대출 승인 등 완전 자동화가 어려움

Propy

스마트 컨트랙트를 활용한
부동산 거래

활용 방안

아직 활성화되지 않은 분야와 그 이유

1 smart contract

2 smart contract & database

3 application

보험

현실 적용이 어려움
- 복잡한 법적 해설 & 예외 상황
- 계약 조건이 정확한 데이터 입력을 전제로 함
- 사고나 질병을 데이터만으로 판단하기 어려움

공급망 관리

(이론) 모든 거래 과정을 블록체인에 기록하면 투명성이 높아짐

데이터를 블록체인과 연결하는 것이 어려움
- SCM을 블록체인과 통합하는 인프라 부족
- ERP시스템과의 연동이 어려움

부동산 거래

중개인을 줄이고 거래 비용 절감 가능
법적 분제와 기존 등기 시스템과의 통합이 어려움
- 법적으로 블록체인 계약이 인정되는지 확실치 않음
- 부동산 등기 시스템이 기존 정부 DB와 연결되어 있어 통합이 어려움
- 가격 협상, 대출 승인 등 완전 자동화가 어려움

활성화되지 못한 주요 원인은
현실 데이터와 블록체인을 연결하는 어려움,
법적 규제, 기존 시스템과의 연동 문제 때문

Etherisc

날씨 기반 보험
비가 오면 자동 보험금 지급

IBM Food Trust

월마트, 네슬레 등이 참여
식품 공급망을 블록체인으로 관리

Propy

스마트 컨트랙트를 활용한
부동산 거래

감사합니다